Leveling Up Your Tradecraft – How to Bolster Your Toolkit

C:\Windows\System32\whoami.exe

- Matthew Eidelberg – Engineering Fellow at **Optiv**
  - Adversarial Simulation Services Lead
  - Author of several Open-Source Tools and Frameworks
  - Research focus – EDR, Evasion, and Bypasses
  - Microsoft Hall of Famer – Related to COM Bypasses
  - Spoken at Defcon Red Team Village & Adversary Village, DerbyCon, BSides LV, GrrCon, RSA and more

Agenda

- Understanding Detection
- OPSEC Consideration
- Demystifying EDR Machine Learning
- New Techniques
- How to Defend Against Them

Payload Development

- Is a lot harder than it looks
- My personal belief:
  - *“Your best offense is to UNDERSTAND the defense”*
- We need to understand what we are trying to defeat in order to be successful

Endpoint Detection Response/Protection

- Typically rely on for detection:
  - Userland Hooking*
  - Kernel Callbacks
  - ETW Events
  - “Machine Learning AI”

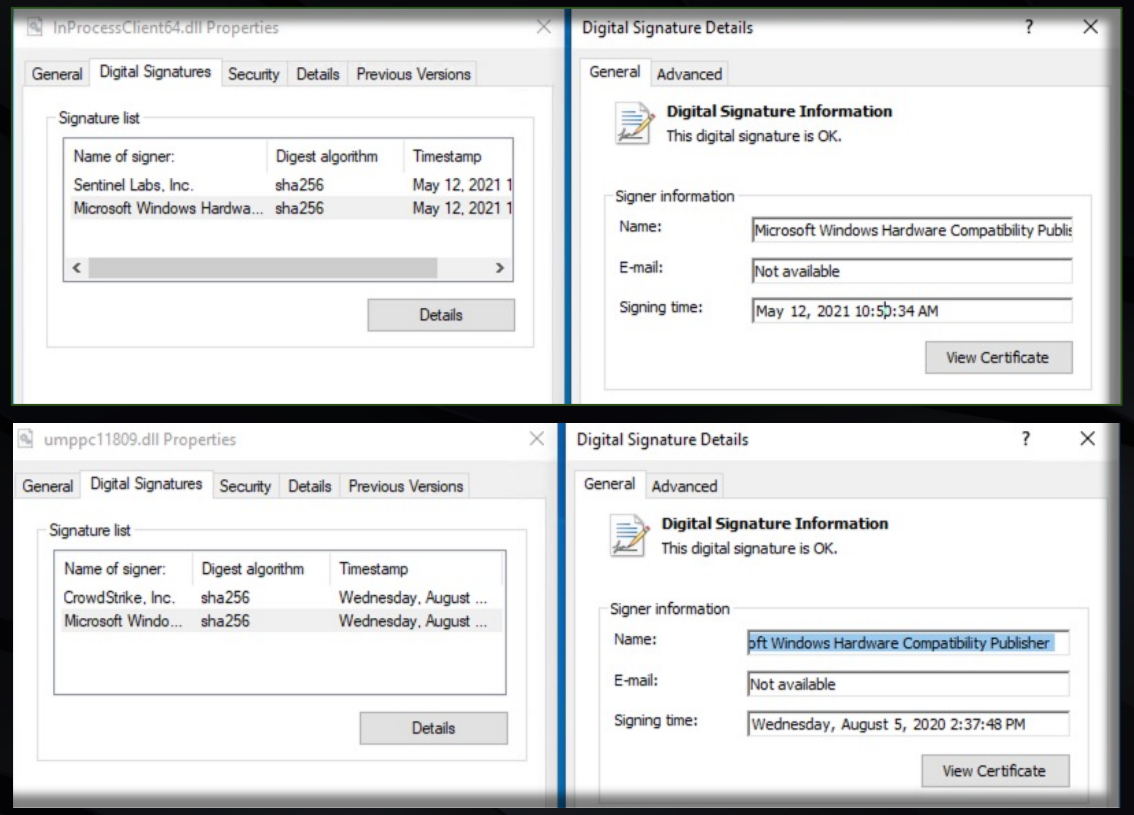
* A lot of the major vendors use Userland Hooking

What We Deploy to Get Around

- Process Injection
- Undocumented API calls for Execution
- BlockDLLs
- Custom Syscalls
- Advanced Evasion Techniques

Their Responses

- Microsoft Code Signed DLLs
  - Vendors are getting their EDR's DLL signed under Microsoft's CA
- Whitelisting Controls
  - Allow only specific executable to run
- "XDR" Controls
  - Gathers telemetry from sources outside of the endpoint



OPSEC: Our Response to Their Response

Microsoft Code Signed DLLs

- Avoid using blockdlls for your initial foothold post-ex when against products like CrowdStrike or Sentinel One

Whitelisting Controls

- Utilizing natively allowed processes to load payloads

“XDR” Controls

- Blocking ETW from writing events

How Do We Get Better

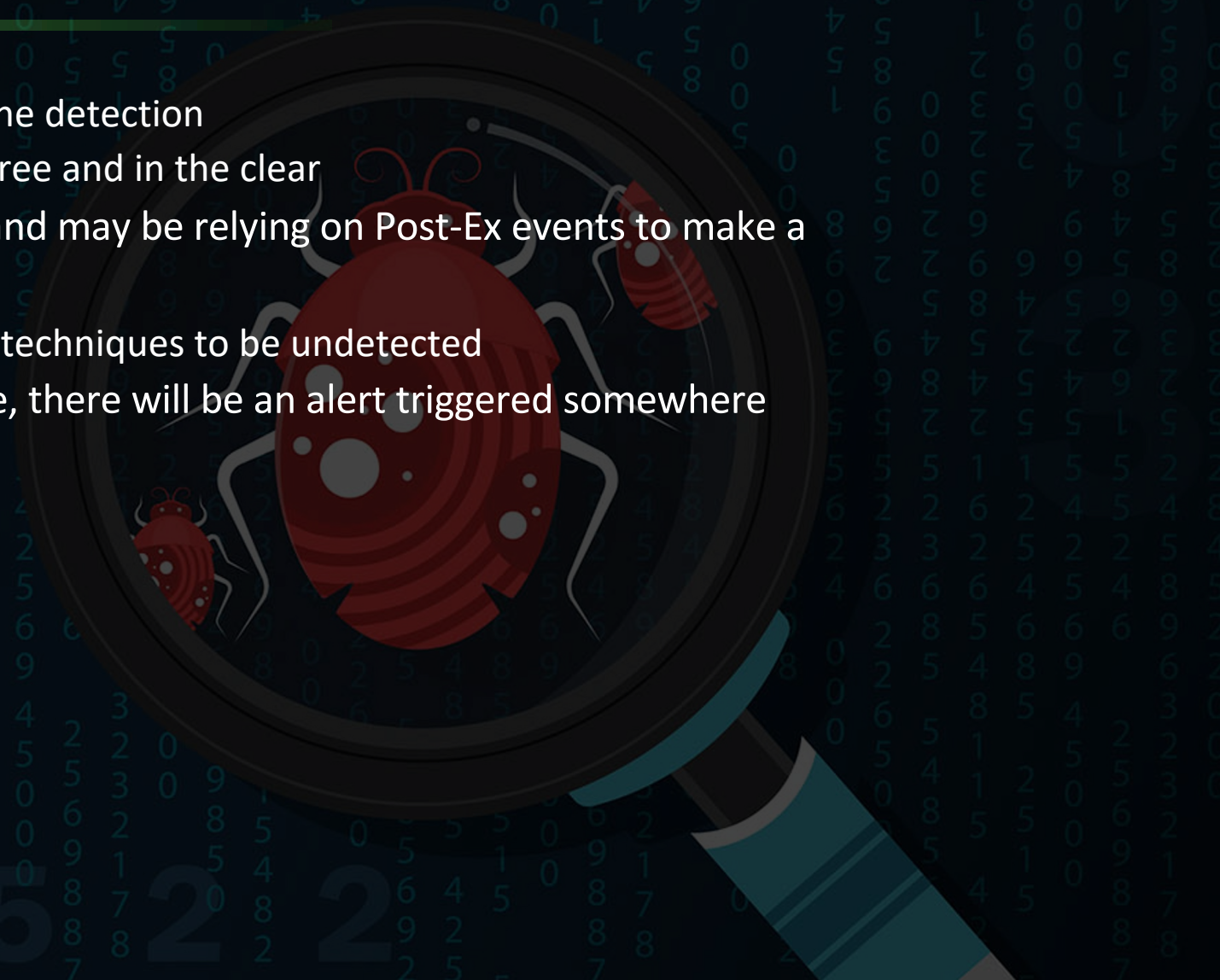
- “Learn what is ACTUALLY triggering the alert”
  - Often, it’s something simple to fix or avoid
- Implants often get the blame when they weren’t even executed yet
- To improve your tradecraft, you must understand what triggered the detection in the first place
- Need to understand the execution chain

Execution Chain

- Method of delivery, or executing the loader on the endpoint
- File containing the implant and other techniques such as anti-sandboxing, unhooking, and decryption mechanisms
- Shellcode or DLL that runs in memory, responsible for establishing the C2 connection

Behavioral Indicators

- Implants aren’t always the primary cause of the detection
- Establishing a foothold doesn’t mean you’re free and in the clear
  - EDRs may not have enough telemetry and may be relying on Post-Ex events to make a determination
- It doesn’t matter if the implant has the latest techniques to be undetected
  - If you're using a documented technique, there will be an alert triggered somewhere
- “Fruit from the poison tree”



OPSEC: Implant Considerations

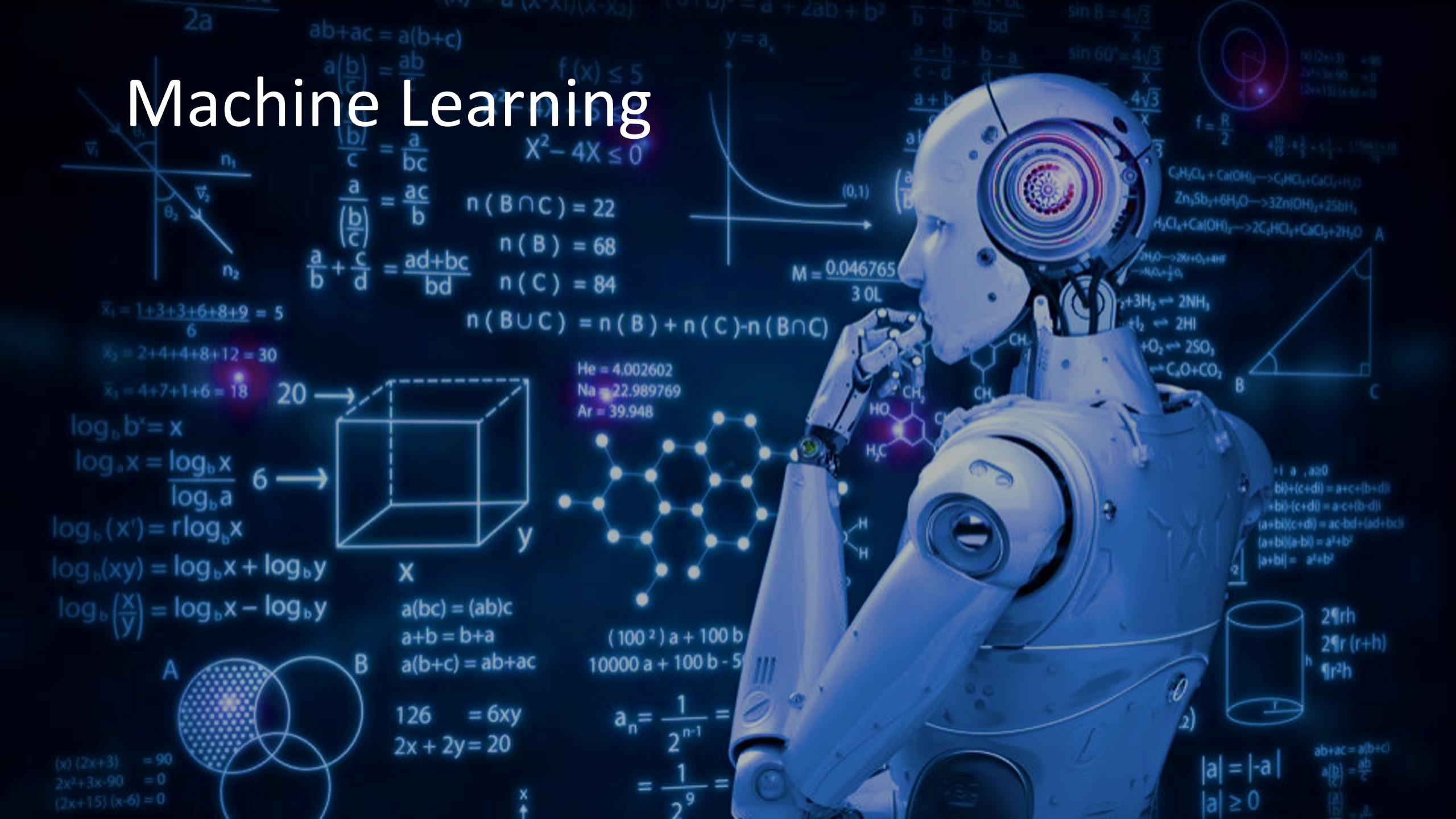
- Use some sort of encryption
- Obfuscation is good but it's no substitute for encryption
- Avoid static values or strings that are massively long
  - Hint: Yara rules are great to learn what blue teams look for
- Using uncommon languages can add a layer of obfuscation

```
strings:
    // 0x10001778 8b 45 08  mov     eax, dword ptr [ebp + 8]
    // 0x1000177b 35 dd 79 19 ae    xor     eax, 0xae1979dd
    // 0x10001780 33 c9     xor     ecx, ecx
    // 0x10001782 8b 55 08  mov     edx, dword ptr [ebp + 8]
    // 0x10001785 89 02     mov     dword ptr [edx], eax
    // 0x10001787 89 ?? ??  mov     dword ptr [edx + 4], ecx
    $op1 = { 8b 45 08 35 dd 79 19 ae 33 c9 8b 55 08 89 02 89 }
    // 0x10002045 74 36     je      0x1000207d
    // 0x10002047 8b 7f 08  mov     edi, dword ptr [edi + 8]
    // 0x1000204a 83 ff 00  cmp     edi, 0
    // 0x1000204d 74 2e     je      0x1000207d
    // 0x1000204f 0f b7 1f  movzx   ebx, word ptr [edi]
    // 0x10002052 8b 7f 04  mov     edi, dword ptr [edi + 4]
    $op2 = { 74 36 8b 7f 08 83 ff 00 74 2e 0f b7 1f 8b 7f 04 }
    // 0x100020cf 74 70     je      0x10002141
    // 0x100020d1 81 78 05 8d 54 24 04      cmp     dword ptr [eax + 5], 0x424548d
    // 0x100020d8 75 1b     jne     0x100020f5
    // 0x100020da 81 78 08 04 cd ?? ??      cmp     dword ptr [eax + 8], 0xc22ecd04
    $op3 = { 74 70 81 78 05 8d 54 24 04 75 1b 81 78 08 04 cd }
```

OPSEC: Loader Consideration

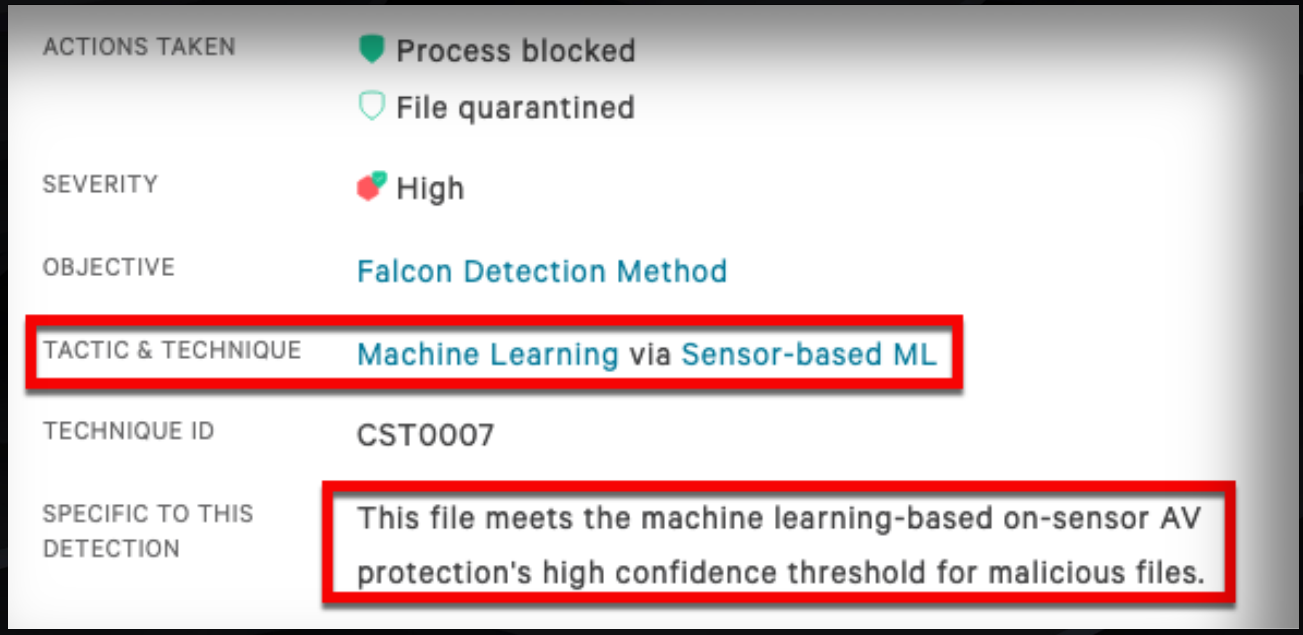
Highly dependent on the environment you're targeting	• What controls are in place
Be mindful of:	• File name • Attributes and metadata • How it looks on disk
Binaries are easy to detect	• Tons of different file extensions that can be used
Avoid large obfuscation strings	• Base64 or XOR are not encryption • Long strings are quite suspicious

Machine Learning

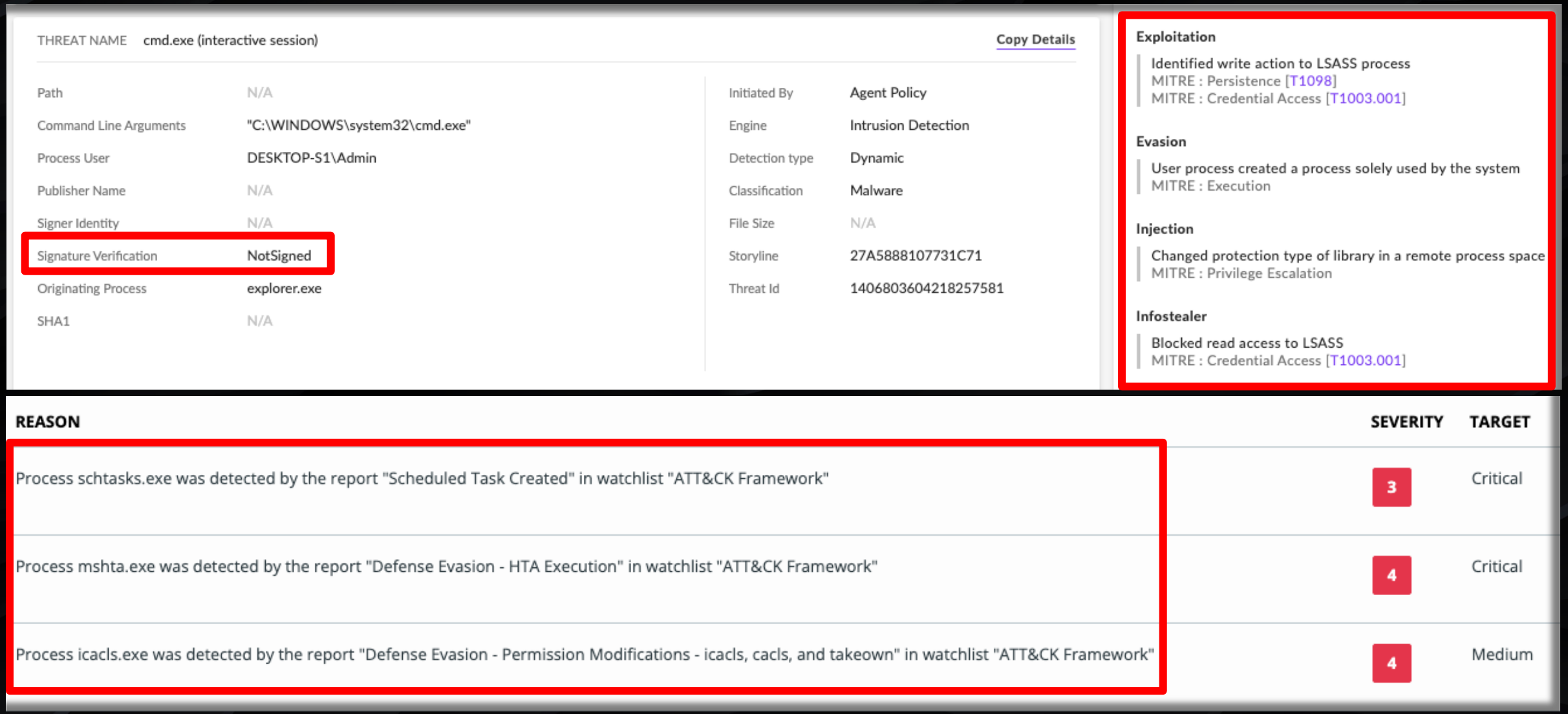


Breaking It Down

- Machine Learning is often an EDR's secret sauce
- So, we can't ever fully know what it's picking up on, and that's the problem
- So long as the alert is there the product is doing its job
- Truth is, you don't know where the gaps are



Actual Behavioral Alerts



THREAT NAME cmd.exe (interactive session) — Copy Details

Path	N/A
Command Line Arguments	"C:\WINDOWS\system32\cmd.exe"
Process User	DESKTOP-S1\Admin
Publisher Name	N/A
Signer Identity	N/A
Signature Verification	NotSigned
Originating Process	explorer.exe
SHA1	N/A
Initiated By	Agent Policy
Engine	Intrusion Detection
Detection type	Dynamic
Classification	Malware
File Size	N/A
Storyline	27A5888107731C71
Threat Id	1406803604218257581

Exploitation

Identified write action to LSASS process
MITRE : Persistence [T1098]
MITRE : Credential Access [T1003.001]

Evasion

User process created a process solely used by the system
MITRE : Execution

Injection

Changed protection type of library in a remote process space
MITRE : Privilege Escalation

Infostealer

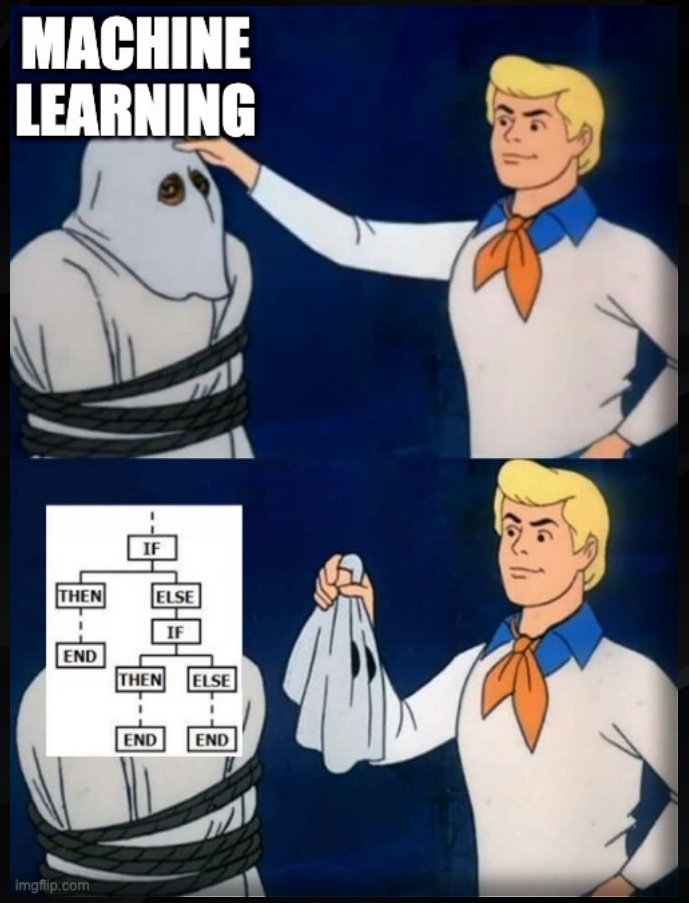
Blocked read access to LSASS
MITRE : Credential Access [T1003.001]

REASON	SEVERITY	TARGET
Process schtasks.exe was detected by the report "Scheduled Task Created" in watchlist "ATT&CK Framework"	3	Critical
Process mshta.exe was detected by the report "Defense Evasion - HTA Execution" in watchlist "ATT&CK Framework"	4	Critical
Process icacls.exe was detected by the report "Defense Evasion - Permission Modifications - icacls, cacls, and takeown" in watchlist "ATT&CK Framework"	4	Medium

Machine Learning – What it really is

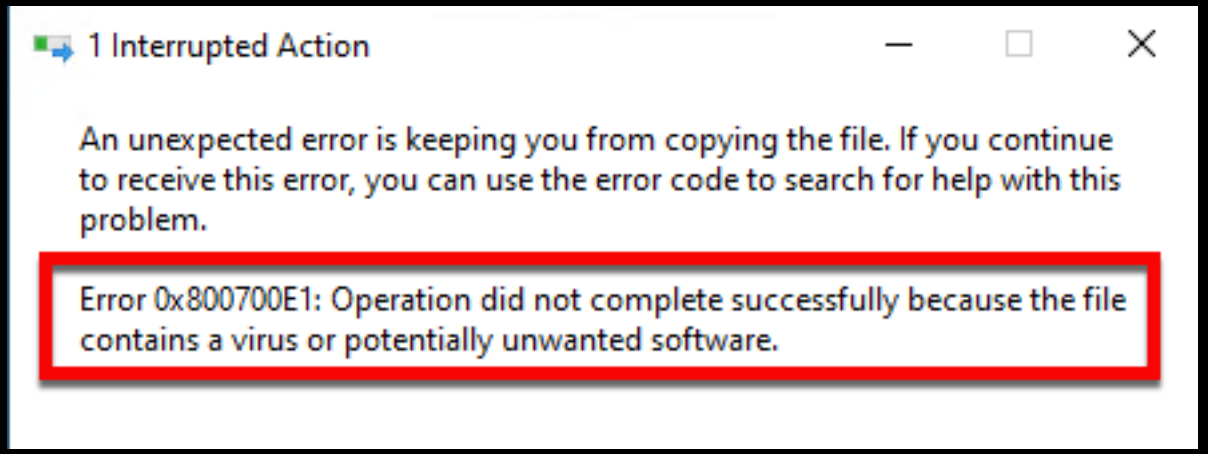
- On-Disk Scanning
- Looking for Strings
- Pre-Runtime scanning
- Checks behavioral patterns **
  - Some even copy and execute in a container

- ** We'll talk more about that



Example Alert: Copying a file

- No Runtime events were performed:
  - Unhooking
  - Module Stomping
  - Thread Spoofing
- Why did this happen?
  - The file contained something they could alert on



Live Field Test

- On a recent Purple team, we performed against a Kernel-level EDR
  - PE file implant stored in a JScript file couldn’t be copied to disk without this alert
  - Took the same file and wrapped it in Mshta handles
  - Used Mshta <url>
  - Execution worked flawlessly
- Why?
  - Mshta pulls it into memory before executing it, avoiding the static disk scanners
    - Static Signatures were never scanned for in memory

Static Signatures

Static Signatures AKA Strings

- Dropping a file to disk and it instantly gets flagged?
  - Very little Runtime events have occurred
- Very effective for stopping any unwanted malware before it has a chance to run
- String based detection is a major go-to for detecting Golang or even Rust payloads

```
rule APT34_VALUEVAULT: apt34 infostealer winmalware
{
    meta:
        Description= "Information stealing malware used by APT34, writt
        Reference = "https://www.fireeye.com/blog/threat-research/2019/

    strings:
        $fsociety = "fsociety.dat" ascii

        $powershell = "New-Object -ComObject Shell.Application" ascii

        $gobuild = "Go build ID: " ascii
```

Downside

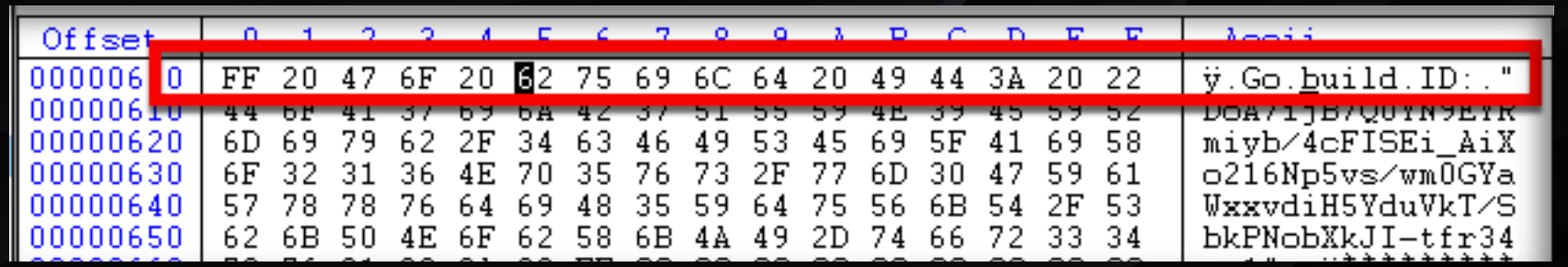
- Because of obfuscation, detections rules must relay on things that can’t be modified
- Creates false positives if not done right
- Our test cases will be two binaries generated from an old Golang framework

PE Manipulation

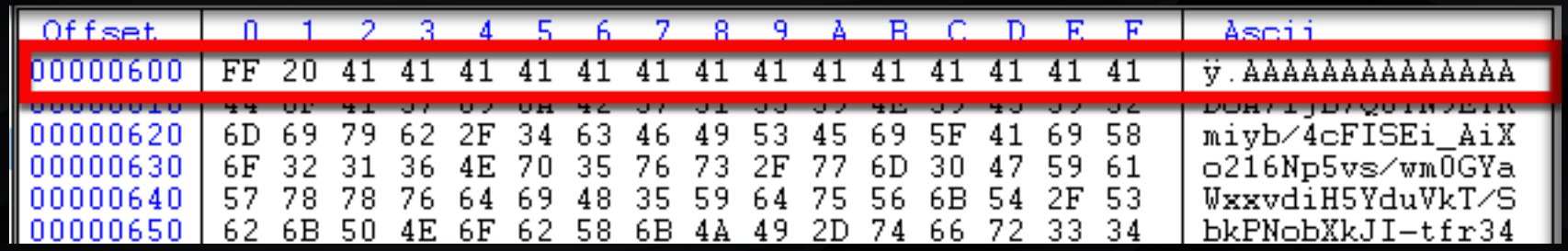
- If we have a sense of some common strings to avoid, we can just strip them out of the file, right?
- The Answer is: Yes, but the real question is, how?
- Change attributes of a PE file that result in the following:
  - An increase to a section size
  - A decrease to a section size
  - Changing a referenced function or library
- Can cause a fault or a PE mis-alignment

Considerations

- We must edit the EXACT bytes (nothing less, nothing more)
- We can’t change any function names



Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	Ascii
000006	0	FF	20	47	6F	20	62	75	69	6C	64	20	49	44	3A	20	22	ÿ.Go.build.ID:."
00000620		6D	69	79	62	2F	34	63	46	49	53	45	69	5F	41	69	58	miyb/4cFISEi_AiX
00000630		6F	32	31	36	4E	70	35	76	73	2F	77	6D	30	47	59	61	o216Np5vs/wm0GYa
00000640		57	78	78	76	64	69	48	35	59	64	75	56	6B	54	2F	53	WxxvdiH5YduVkT/S
00000650		62	6B	50	4E	6F	62	58	6B	4A	49	2D	74	66	72	33	34	bkPNobXkJI-tfr34



Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	Ascii
00000600		FF	20	41	41	41	41	41	41	41	41	41	41	41	41	41	41	ÿ.AAAAAAAAAAAAAA
00000620		6D	69	79	62	2F	34	63	46	49	53	45	69	5F	41	69	58	miyb/4cFISEi_AiX
00000630		6F	32	31	36	4E	70	35	76	73	2F	77	6D	30	47	59	61	o216Np5vs/wm0GYa
00000640		57	78	78	76	64	69	48	35	59	64	75	56	6B	54	2F	53	WxxvdiH5YduVkT/S
00000650		62	6B	50	4E	6F	62	58	6B	4A	49	2D	74	66	72	33	34	bkPNobXkJI-tfr34

Results

- When we test this, we see something different happen
- Rather than the Machine Learning flagging it before, we can run it now
- When we execute it, we see these beacons come in

HAPUBWS *	HAPUBWS-PC	0f8505baf9fcefee3749c4b73b85af7d472e0e1
HAPUBWS *	HAPUBWS-PC	cbdaf29fcba21662e391eb553e8c41972e46948

- Which is the Hash value of our test files:

```
meidelbera@OPT008385 Test Cases % shasum -a 256 code.exe
0f8505baf9fcefee3749c4b73b85af7d472e0e1d3f6f110442a8f348736410e3  code.exe
meidelbera@OPT008385 Test Cases % shasum -a 256 iMKrmHICwlkTmCF.exe
cbdaf29fcba21662e391eb553e8c41972e46948f43fcf14f3cb5e829bf1899d7  iMKrmHICwlkTmCF.exe
meidelberg@OPT008385 Test Cases %
```

What does this mean

- We removed enough of the static indicators on disk that it wasn't picked up
- This doesn't mean that we're safe, but it does show that by removing these strings, CrowdStrike had to rely on other Telemetry (at Runtime) to see something suspicious

- This also means that our other evasion techniques can now come into play:
  - AMSI & ETW Patching
  - Unhooking EDR
  - Sandboxing
  - Module Stomping
  - Thread Spoofing

Inflate Mode

- Pretty much all EDRs can’t scan both on disk or in memory files beyond a certain size
  - The size is usually north of 100mb
- Just a few lines

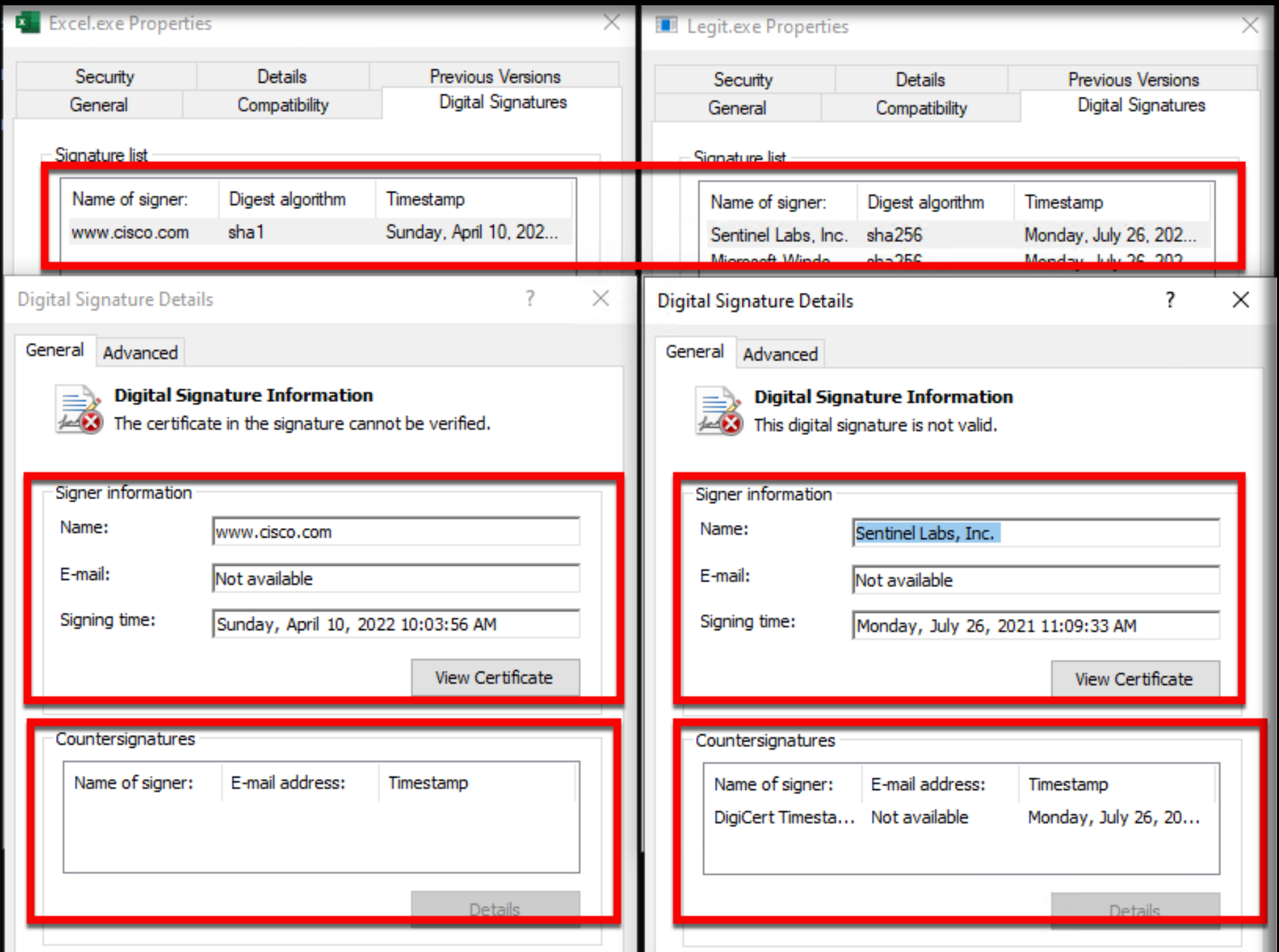
```
func Padding(buff []byte, size int) []byte {
    str1 := "0"
    res1 := strings.Repeat(str1, (size * 1024 * 1024))
    sum := string(buff) + res1
    mydata := []byte(sum)
    return mydata

}
```

New Tool: Mangle

- Strips out known strings related to Golang
- Clones the actual code-signed file
  - This copies all attributes of the certificate, including the chains
  - This is different than Limelighter which takes a fake cert using a domain
- Inflates a payload to any size you want by padding the PE file with zeros

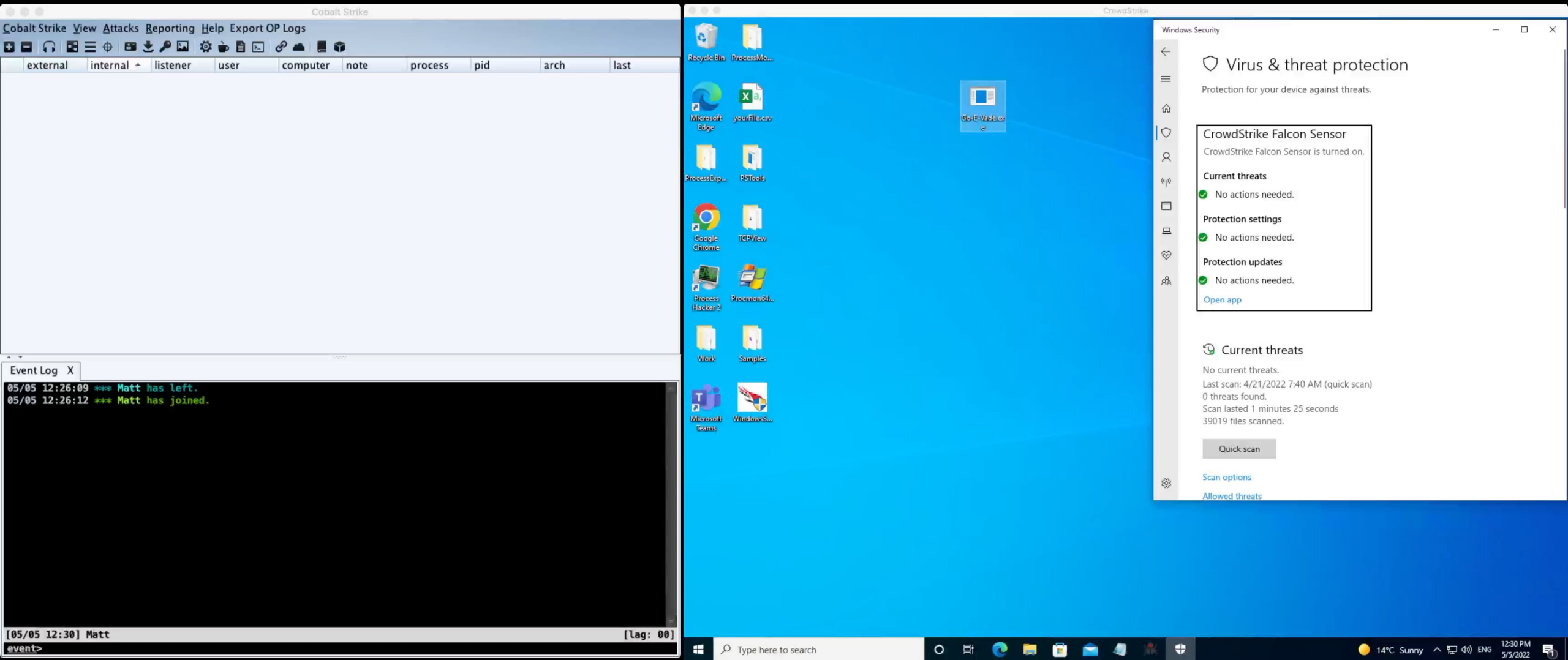
Cloning Certificate Chains

- Copies the entire code signing certificate chain from a legitimate DLL
- This includes the full cert chain
- Limelighter historically created a fake cert using a Domain value to sign the executable


Excel.exe Properties
Security
Details
Previous Versions
General
Compatibility
Digital Signatures
Signature list
Name of signer:
Digest algorithm
Timestamp
www.cisco.com
sha1
Sunday, April 10, 202...
Digital Signature Details
General
Advanced
Digital Signature Information
The certificate in the signature cannot be verified.
Signer information
Name:
www.cisco.com
E-mail:
Not available
Signing time:
Sunday, April 10, 2022 10:03:56 AM
View Certificate
Countersignatures
Name of signer:
E-mail address:
Timestamp
Details

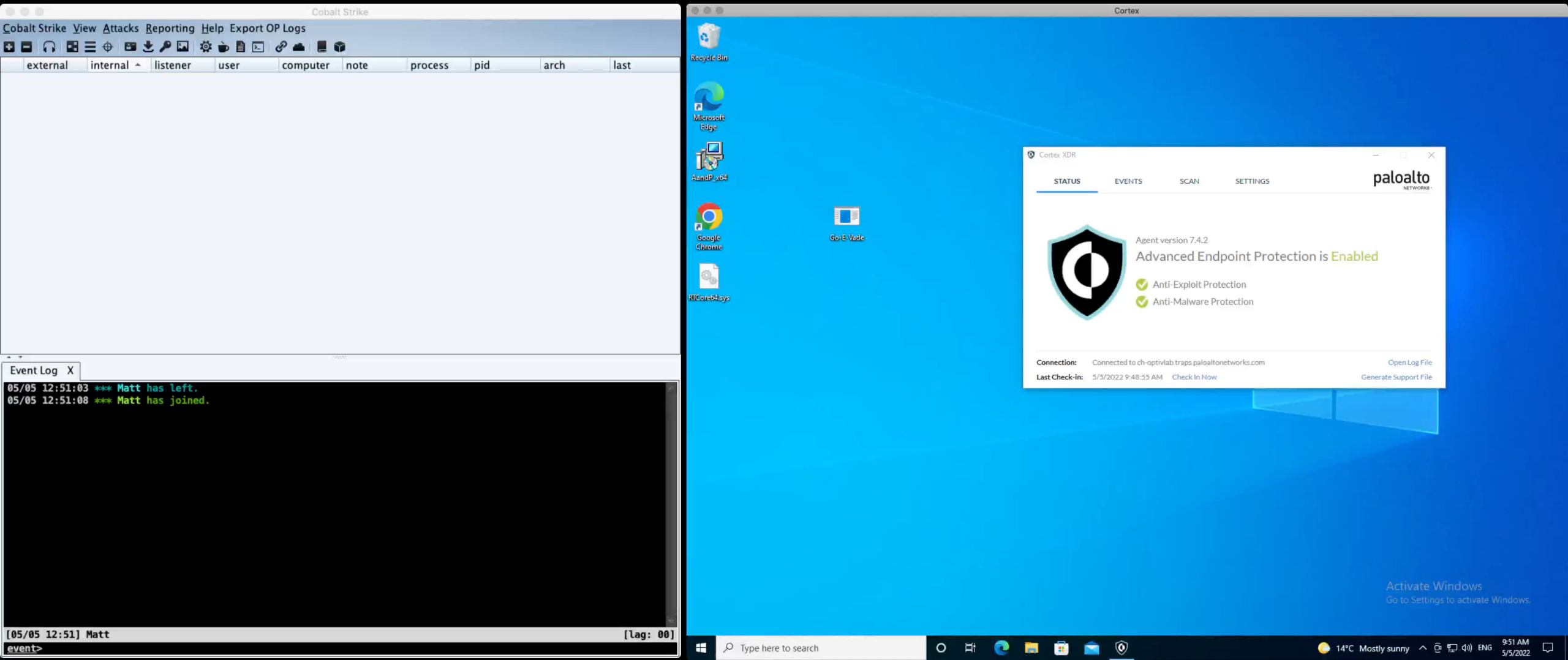
Legit.exe Properties
Security
Details
Previous Versions
General
Compatibility
Digital Signatures
Signature list
Name of signer:
Digest algorithm
Timestamp
Sentinel Labs, Inc.
sha256
Monday, July 26, 202...
Digital Signature Details
General
Advanced
Digital Signature Information
This digital signature is not valid.
Signer information
Name:
Sentinel Labs, Inc.
E-mail:
Not available
Signing time:
Monday, July 26, 2021 11:09:33 AM
View Certificate
Countersignatures
Name of signer:
E-mail address:
Timestamp
DigiCert Timesta...
Not available
Monday, July 26, 20...
Details

Demo


Cobalt Strike
Cobalt Strike View Attacks Reporting Help Export OP Logs
external internal listener user computer note process pid arch last
Event Log X
05/05 12:26:09 *** Matt has left.
05/05 12:26:12 *** Matt has joined.
[05/05 12:30] Matt [lag: 00]
event>
CrowdStrike
Recycle Bin
ProcessMo...
Microsoft Edge
yourFile.csv
Go-E-Vade.exe
ProcessExp...
PSTools
Google Chrome
TCPView
Process Hacker 2
Procmon64...
Work
Samples
Microsoft Teams
WindowsS...
Windows Security
Virus & threat protection
Protection for your device against threats.
CrowdStrike Falcon Sensor
CrowdStrike Falcon Sensor is turned on.
Current threats
No actions needed.
Protection settings
No actions needed.
Protection updates
No actions needed.
Open app
Current threats
No current threats.
Last scan: 4/21/2022 7:40 AM (quick scan)
0 threats found.
Scan lasted 1 minutes 25 seconds
39019 files scanned.
Quick scan
Scan options
Allowed threats
Type here to search
14°C Sunny
ENG
12:30 PM
5/5/2022

Fun Fact

- This technique also bypasses Kernel EDRs

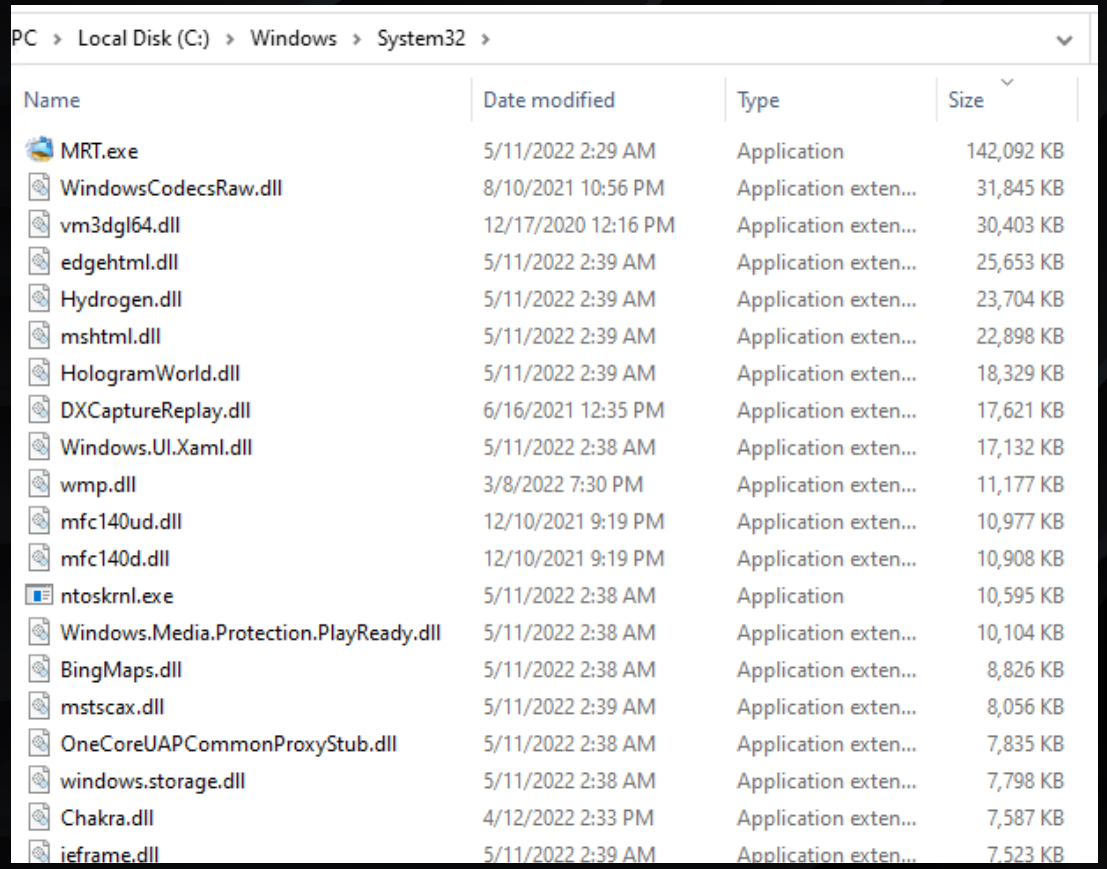

Cobalt Strike
Cobalt Strike View Attacks Reporting Help Export OP Logs
external internal listener user computer note process pid arch last
Event Log X
05/05 12:51:03 *** Matt has left.
05/05 12:51:08 *** Matt has joined.
[05/05 12:51] Matt [lag: 00]
event>
Cortex
Recycle Bin
Microsoft Edge
AandP_x64
Google Chrome
Go-E-Vade
RTCore64.sys
Cortex XDR
STATUS EVENTS SCAN SETTINGS
paloalto NETWORKS
Agent version 7.4.2
Advanced Endpoint Protection is Enabled
Anti-Exploit Protection
Anti-Malware Protection
Connection: Connected to ch-optivlab.traps.paloaltonetworks.com
Open Log File
Last Check-in: 5/5/2022 9:48:55 AM Check In Now
Generate Support File
Activate Windows
Go to Settings to activate Windows.
Type here to search
14°C Mostly sunny
ENG
9:51 AM
5/5/2022

Indicators of Compromise

Indicators

Binaries & DLL are typically not over 30 MB

Any large executables would be found in Program files or System32



Defending Against This

- This isn’t a failure of EDRs
  - EDRs are a key piece of any enterprise’s security stack
  - Depending on them as the sole thing to protect your organization is an unfair expectation
- Implementing targeted rules that look for excessively large files in non-install places is a great start
- Highly tuned Whitelisting controls are key
  - Any loaders dropped to disk or native executables that can used maliciously will be prevented

Questions

- Mangle Github Repo can be found here (https://github.com/optiv/mangle)
- Slides can be found here (https://github.com/Tylous/Slides)
- Twitter: @Tyl0us
- Github: Tylous